Master Course
Computer Networks
IN2097

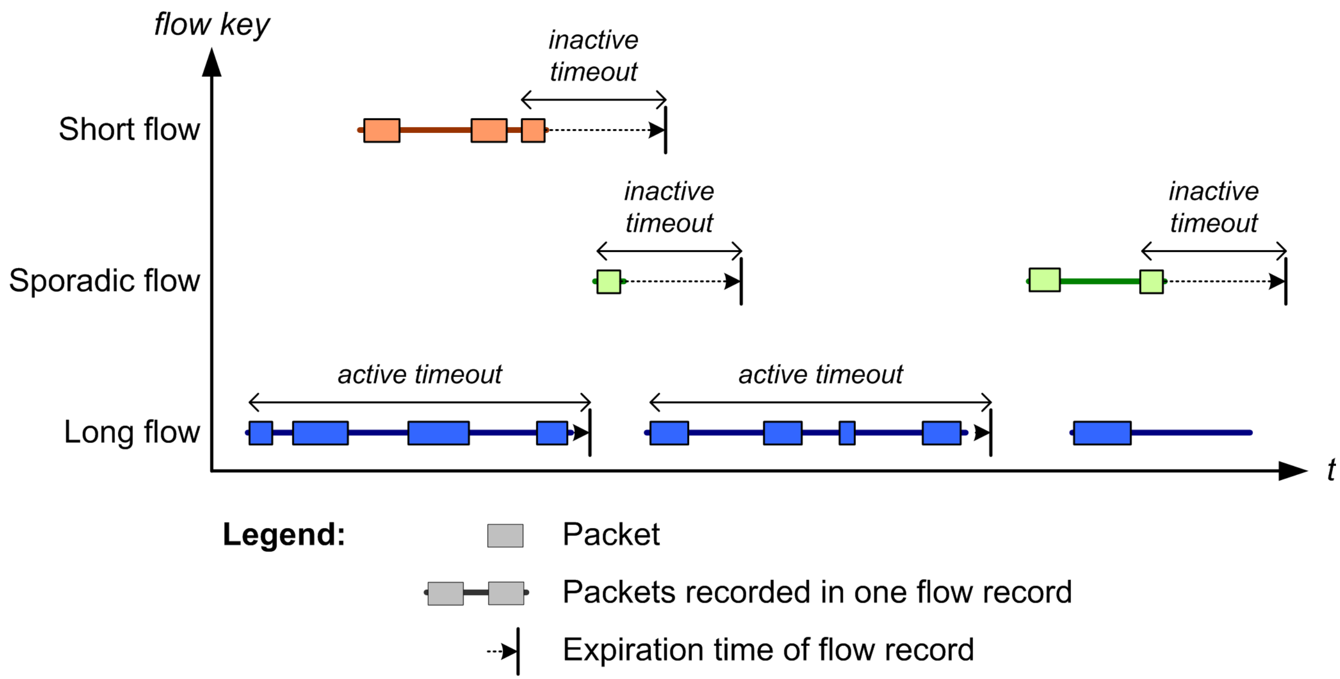
Prof. Dr.-Ing. Georg Carle
Christian Grothoff, Ph.D.
Stephan Günther

Chair for Network Architectures and Services
Department of Computer Science
Technische Universität München
http://www.net.in.tum.de

Chapter: Network Measurements

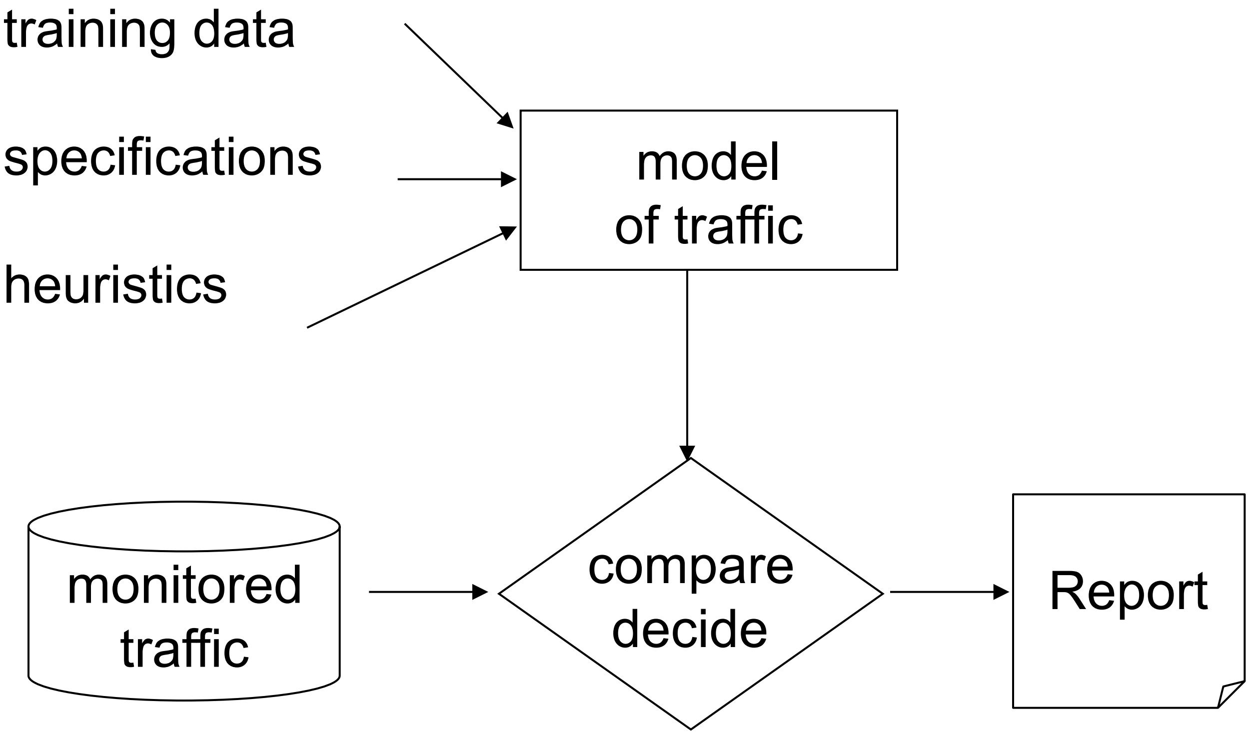
- continuation -

Flow-based Traffic Measurements



Detecting Anomalies

- Modeling phase
  - derive model of normal traffic
- Analysis phase
  - compare observed traffic with normal traffic
  - report significant deviation from normal traffic



Measurement Granularity and Analysis Goals

Granularity	Analysis goals
End System	**Malware Analysis using Honeypots and Sandboxes,** Traffic generation, Analysis of Encrypted Communications
Full Packets	Deep Packet Inspection, **Payload Analysis,** Session Analysis (limited to unencrypted traffic)
Packet Headers	Traceback, Delay Measurements, **Statistical Traffic Classification**
Flows	Host Activity Analysis, Dependency Analysis, **Detection of Scans, Attacks, Traffic Anomalies**
Traffic Aggregates	Accounting, Traffic Matrix Analysis, **Detection of Traffic Anomaly and Network Outage**

- Capturing and analyzing worm and malware traffic

- Packet sampling for signature detection
- Malware and botnet analysis

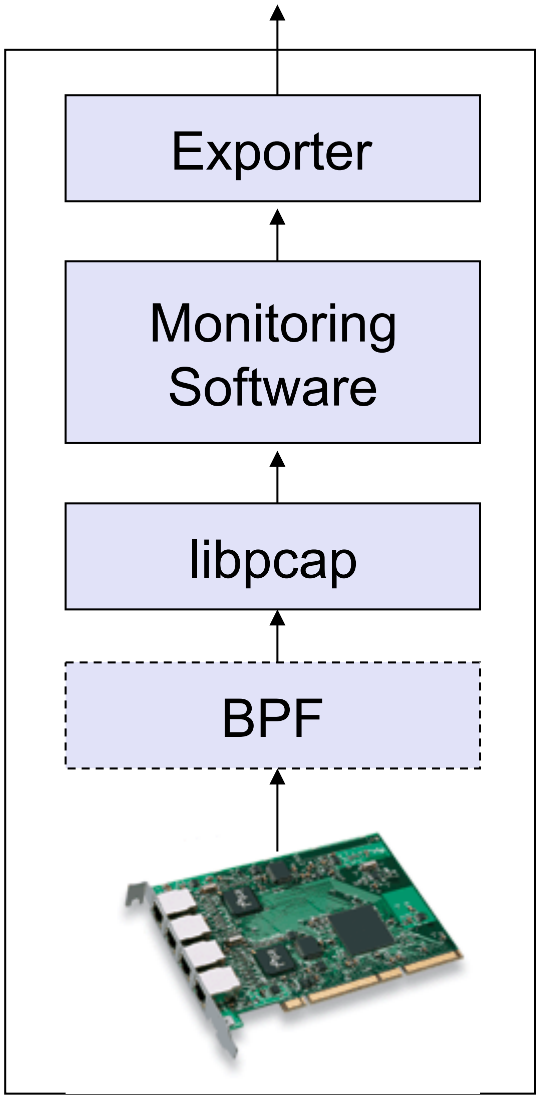
- Machine Learning methods
- Markov models
- Malware and Botnet analysis

- Identification of incidents and anomaly causes

- Control charts
- Time-series analysis
- Principal component analysis
- Wavelet analysis
- Data mining

Monitoring Probe

- Standardized data export
- Monitoring Software
- HW adaptation, [filtering]
- OS dependent interface (e.g. Linux, BSD)
- Network interface



High-Speed Network Monitoring

- Requirements
  - Multi-Gigabit/s Links
  - Cheap hardware and software → standard PC
  - Simple deployment

- Problems
  - Several possible bottlenecks in the path from capturing to final analysis

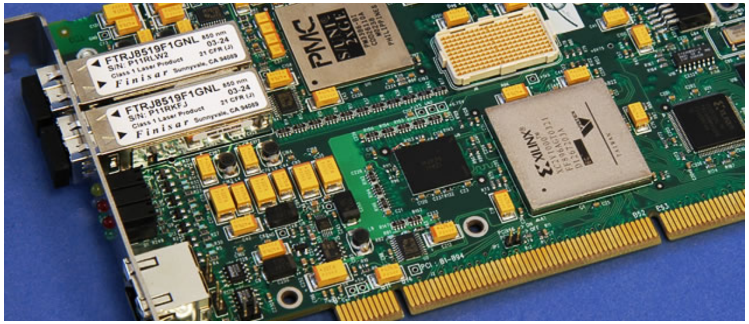
Bottlenecks?

High-Speed Network Monitoring II

- Approaches
  - High-end (intelligent) network adapters
    - Support efficient transfer of packets to main memory
    - Can do filtering, time-stamping etc. on their own
  - Sophisticated algorithms/techniques in OS stack for
    - Maintaining packet queues
    - Elimination of packet copy operations
    - Maintaining state (e.g., managing hash tables describing packet flows; sophisticated packet classification algorithms)
  - Sampling
  - Filtering
  - Aggregation

Special Network Adapters

- Server NICs (Network Interface Cards)
  - Direct access to main memory (without CPU assistance)
  - Processing of multiple packets in a single block (reduction of copy operations)
    → Reduced interrupt rates



- Monitoring interface cards
  - Dedicated monitoring hardware (usually only RX, no TX)
  - Programmable, i.e. certain processing (filtering, high-precision timestamps, ...) can be performed on the network interface card

Memory Management I

- Reduction of copy operations
  - Copy operations can be reduced by only transferring references pointing to memory positions holding the packet
  - Management of the memory is complex, garbage collection required
- Aggregation
  - If aggregated results are sufficient, only counters have to be maintained

Memory Management II

- Hash tables
  - Allow fast access to previously stored information
  - Depending on the requirements, different sections of a packet can be used as input to the hash function
- Multi-dimensional packet classification algorithms
  - Allow to test for 1,000s of complex filtering rules within one lookup operation (e.g., "all TCP packets from network 131.159.14.0/24, but not 131.159.14.0/27, and with source port 80, 443 or 6666–6670, but not with destination address 192.168.69.96–192.168.69.99 ⇨ Apply rule 34")
  - Sophisticated data structures, e.g. tree-based
    ⇨ Lookups fast, but tree alterations costly

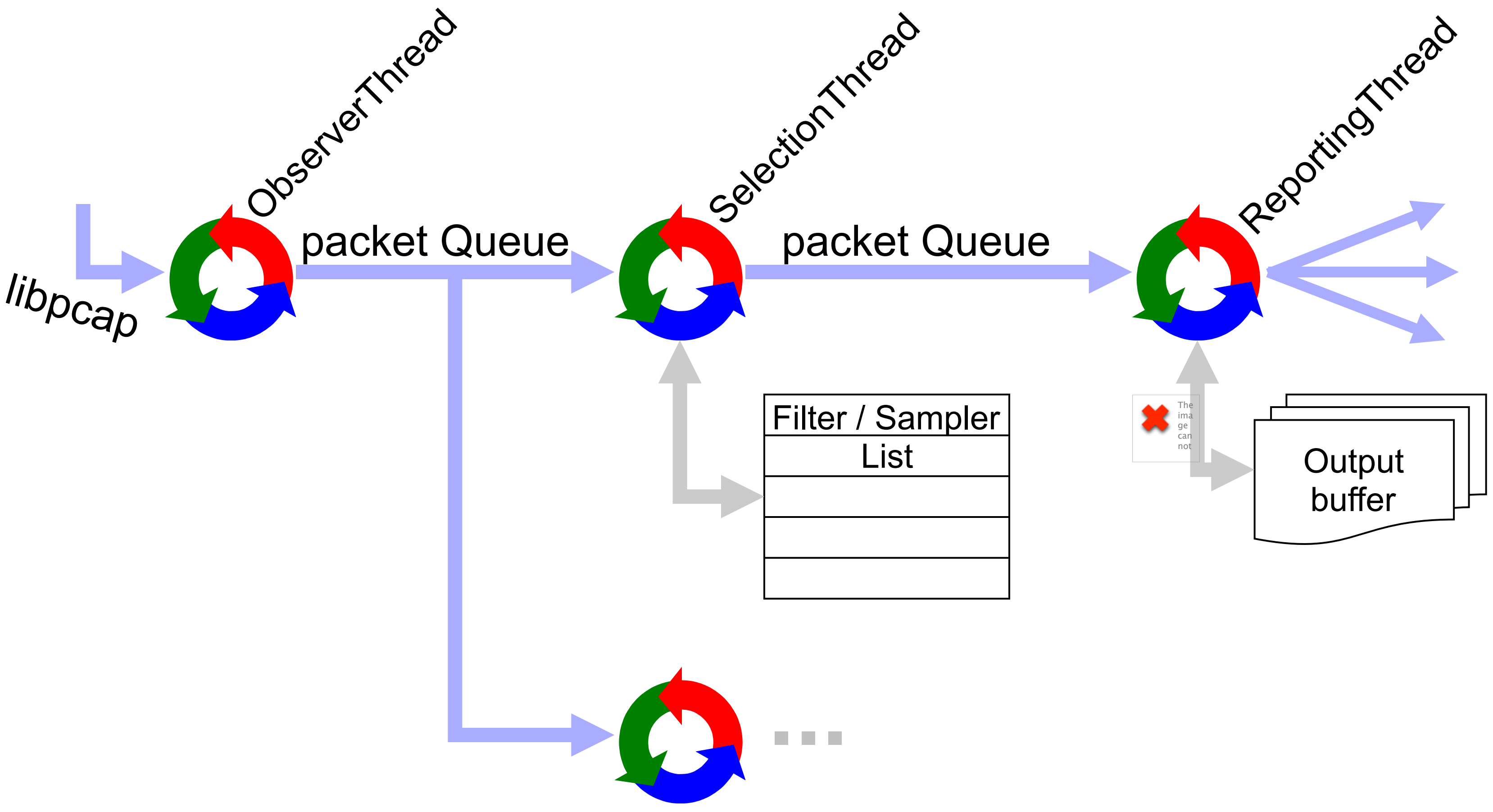
Packet Sampling

- Goals
  - Reduction of the number of packets to analyze
  - Statistically dropping packets
- Sampling algorithms
  - Systematic sampling
    - Periodic selection of every n-th element of a trace
    - Selection of packets arriving at pre-defined points in time
  - Random sampling
    - n-out-of-N
    - Probabilistic
  - "Time machine" sampling: Sample first N bytes of every flow

Packet Filtering

- Goals
  - Reduction of the number of packets to analyze
  - Possibility to look for particular packet flows in more detail, or to completely ignore other packet flows
- Filter algorithms (explained subsequently)
  - Mask/match filtering
  - Router state filtering
  - Hash-based selection

Implementation of PSAMP / IPFIX



- Example: Vermont IPFix software, c.f.
  http://www.history-project.net/index.php?show=vermont
  https://github.com/constcast/vermont/wiki

Cisco Netflow Implementations

- Catalyst 6500 and Cisco 7600
  - EARL6/EARL7 ASICs
  - 128k or 256k NetFlow TCAM
  - hash tables: 16K or 32K rows with 8 elements
  - performance: 30 Mpps
- Superior Engine 720
  - EARL8 ASIC
  - 512k NetFlow TCAM
  - performance: 60 Mpps
- Application Extension Platform (AXP)
  - Intel x86 Processor (Core 2 Duo 1.86GHz)
  - RAM: 512 MB to 4 GB
  - HDD: 2 GB (flash) to 1 TB

Visualisation of Flow Data

- given: machine-readable data

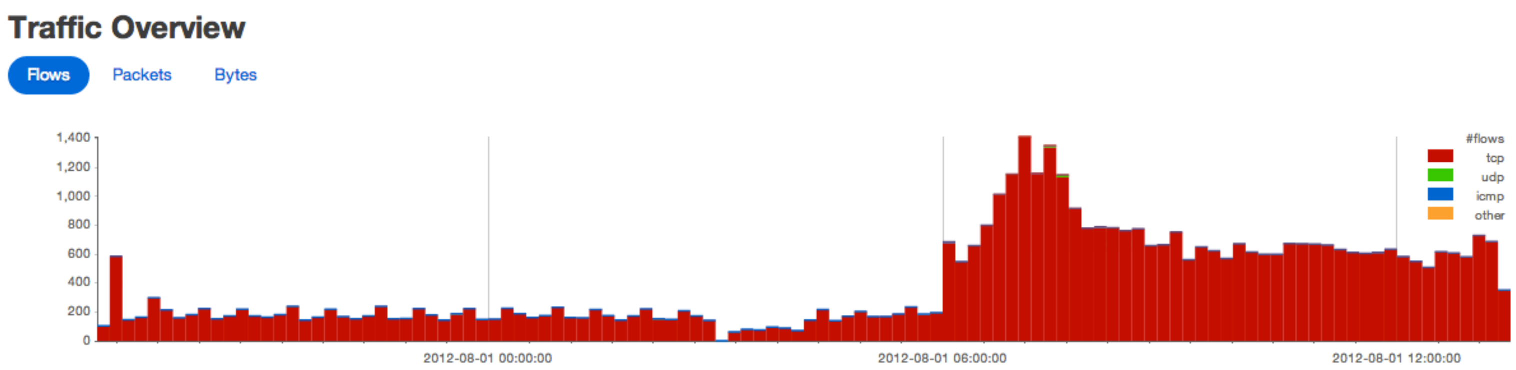
{ "_id" : BinData(0,"MTM0Mzc1MzQwMDIxOTI1Nzc5NTUyMTkzOTc5NDI1MTcwNDE0NDM2"), "bucket" : 1343753400, "bytes" : 367, "dstIP" : NumberLong("2193979425"), "dstPort" : 443, "flows" : 1, "pkts" : 6, "proto" : 6, "srcIP" : NumberLong("2192577955"), "srcPort" : 17041, "tcp" : { "bytes" : 367, "flows" : 1, "pkts" : 6 } }
{ "_id" : BinData(0,"MTM0Mzc1MzQwMDIxOTM5Nzk0MjUyMTkyNTc3OTU1NDQzMTcwNDE2"), "bucket" : 1343753400, "bytes" : 1929, "dstIP" : NumberLong("2192577955"), "dstPort" : 17041, "flows" : 1, "pkts" : 5, "proto" : 6, "srcIP" : NumberLong("2193979425"), "srcPort" : 443, "tcp" : { "bytes" : 1929, "flows" : 1, "pkts" : 5 } }
{ "_id" : BinData(0,"MTM0Mzc1MzQwMDIxOTI1Nzc5NTUyMTkzOTc5NDI1MTcwNDI0NDM2"), "bucket" : 1343753400, "bytes" : 367, "dstIP" : NumberLong("2193979425"), "dstPort" : 443, "flows" : 1, "pkts" : 6, "proto" : 6, "srcIP" : NumberLong("2192577955"), "srcPort" : 17042, "tcp" : { "bytes" : 367, "flows" : 1, "pkts" : 6 } }
{ "_id" : BinData(0,"MTM0Mzc1MzQwMDIxOTM5Nzk0MjUyMTkyNTc3OTU1NDQzMTcwNDI2"), "bucket" : 1343753400, "bytes" : 1929, "dstIP" : NumberLong("2192577955"), "dstPort" : 17042, "flows" : 1, "pkts" : 5, "proto" : 6, "srcIP" : NumberLong("2193979425"), "srcPort" : 443, "tcp" : { "bytes" : 1929, "flows" : 1, "pkts" : 5 } }
{ "_id" : BinData(0,"MTM0Mzc1MzQwMDIxOTI1Nzc5NTUyMTkzOTc5NDI1MTcwNDM0NDM2"), "bucket" : 1343753400, "bytes" : 367, "dstIP" : NumberLong("2193979425"), "dstPort" : 443, "flows" : 1, "pkts" : 6, "proto" : 6, "srcIP" : NumberLong("2192577955"), "srcPort" : 17043, "tcp" : { "bytes" : 367, "flows" : 1, "pkts" : 6 } }

- Goals
  - human-friendly representation
  - graphical presentation of logical relationships

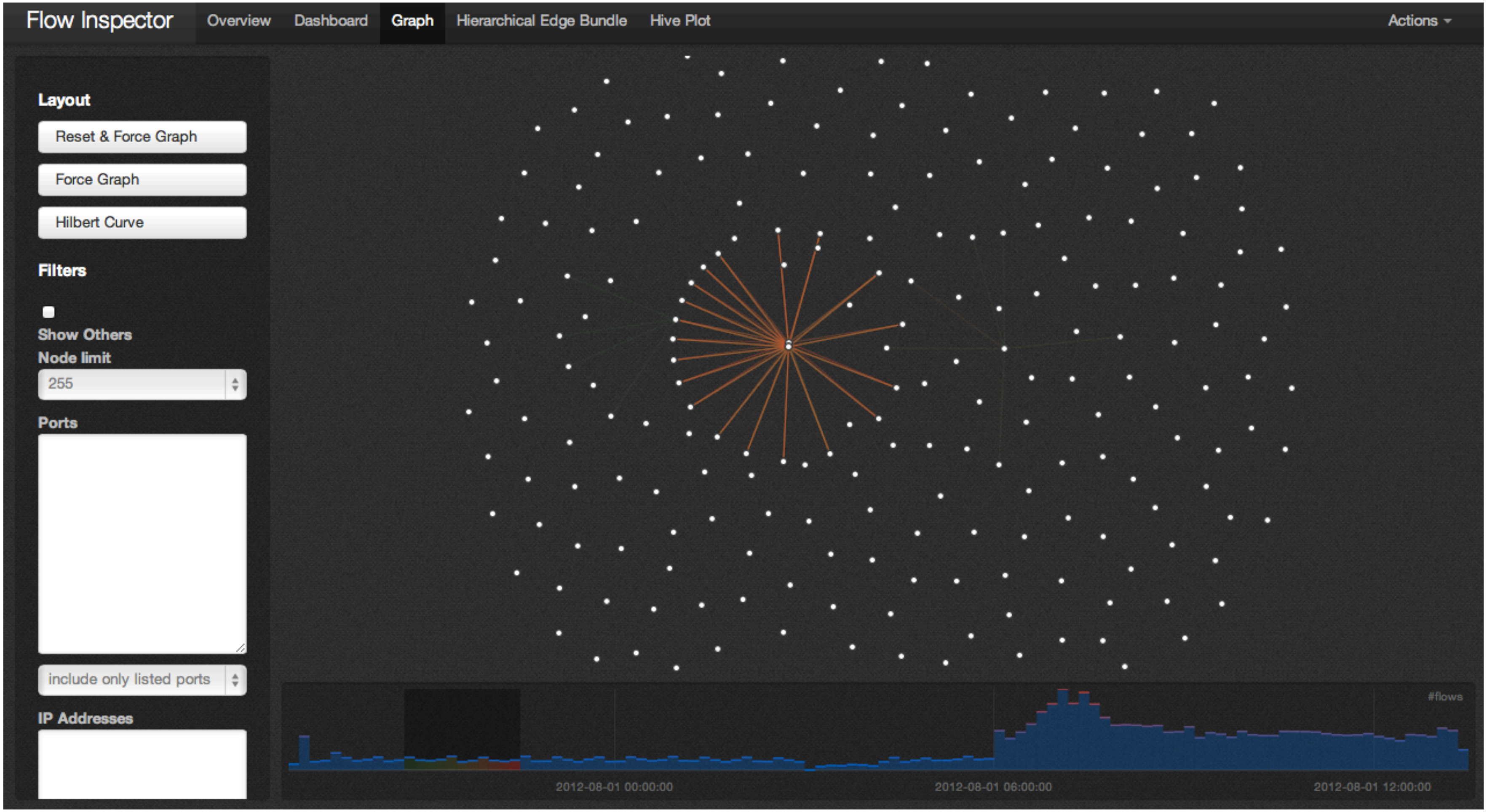
Flow Data Visualisation Examples

- Ordering by hosts

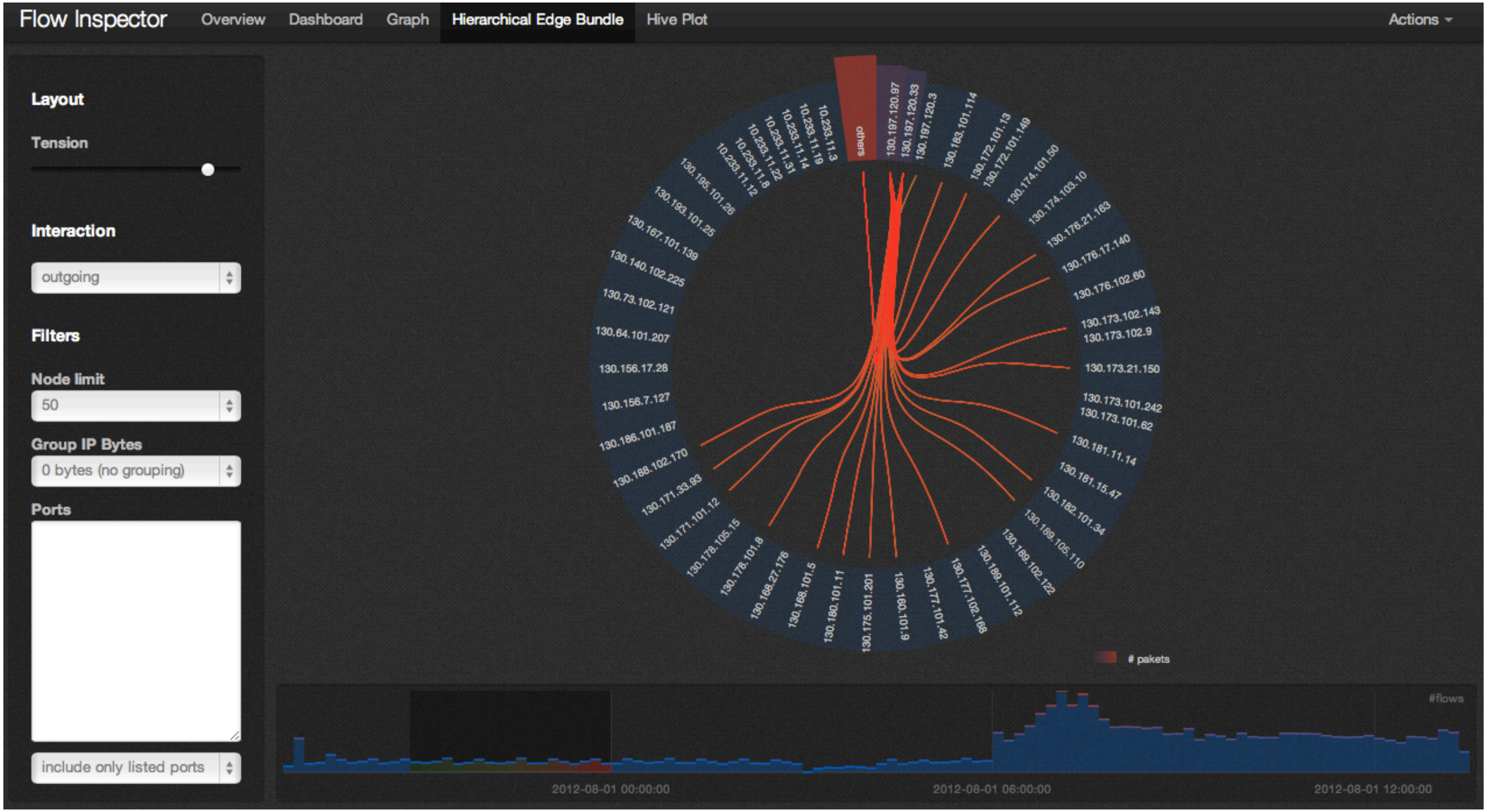
- Traffic over time



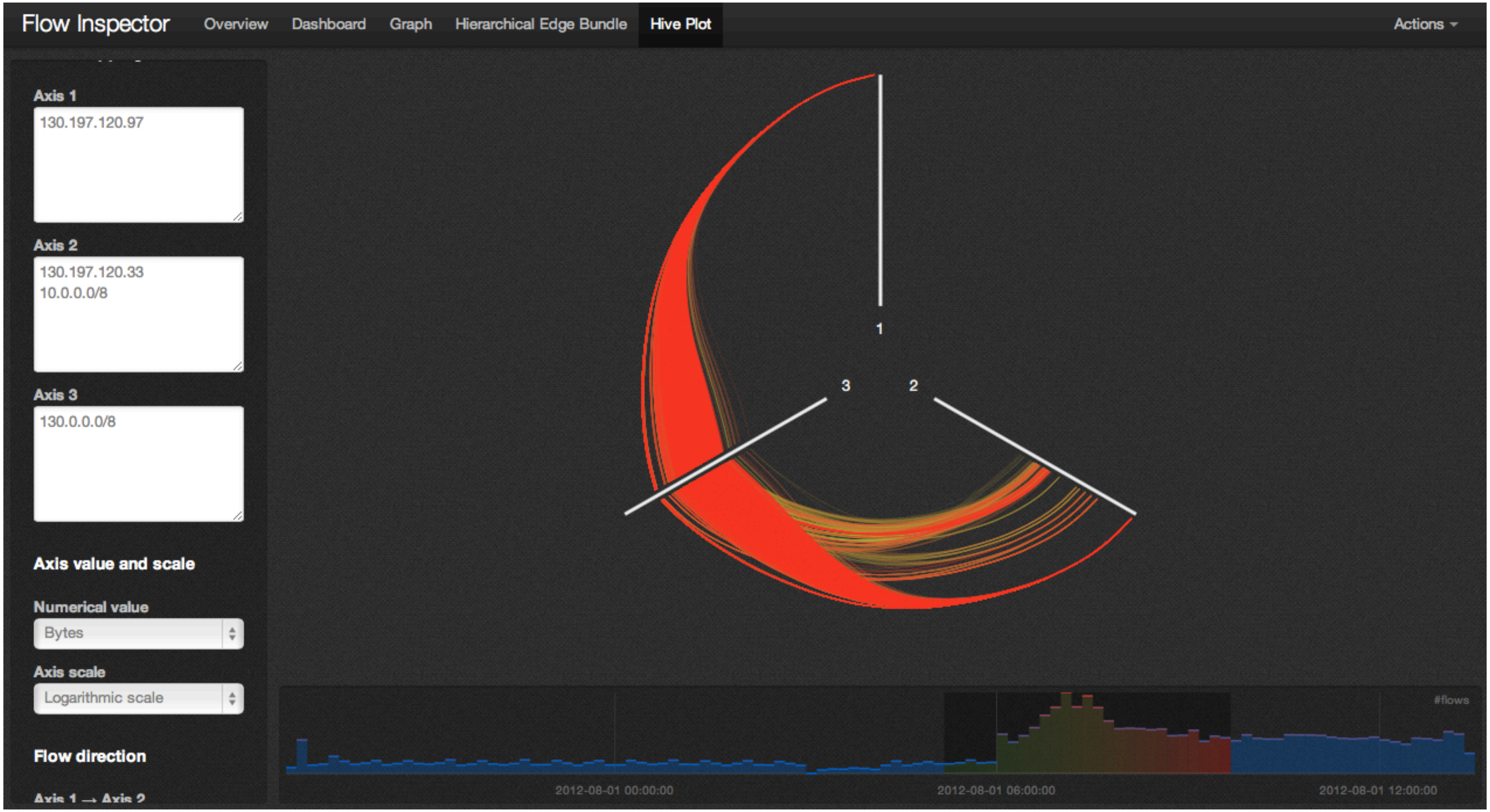
Flow Data Visualisation Examples



Flow Data Visualisation Examples



Flow Data Visualisation Examples



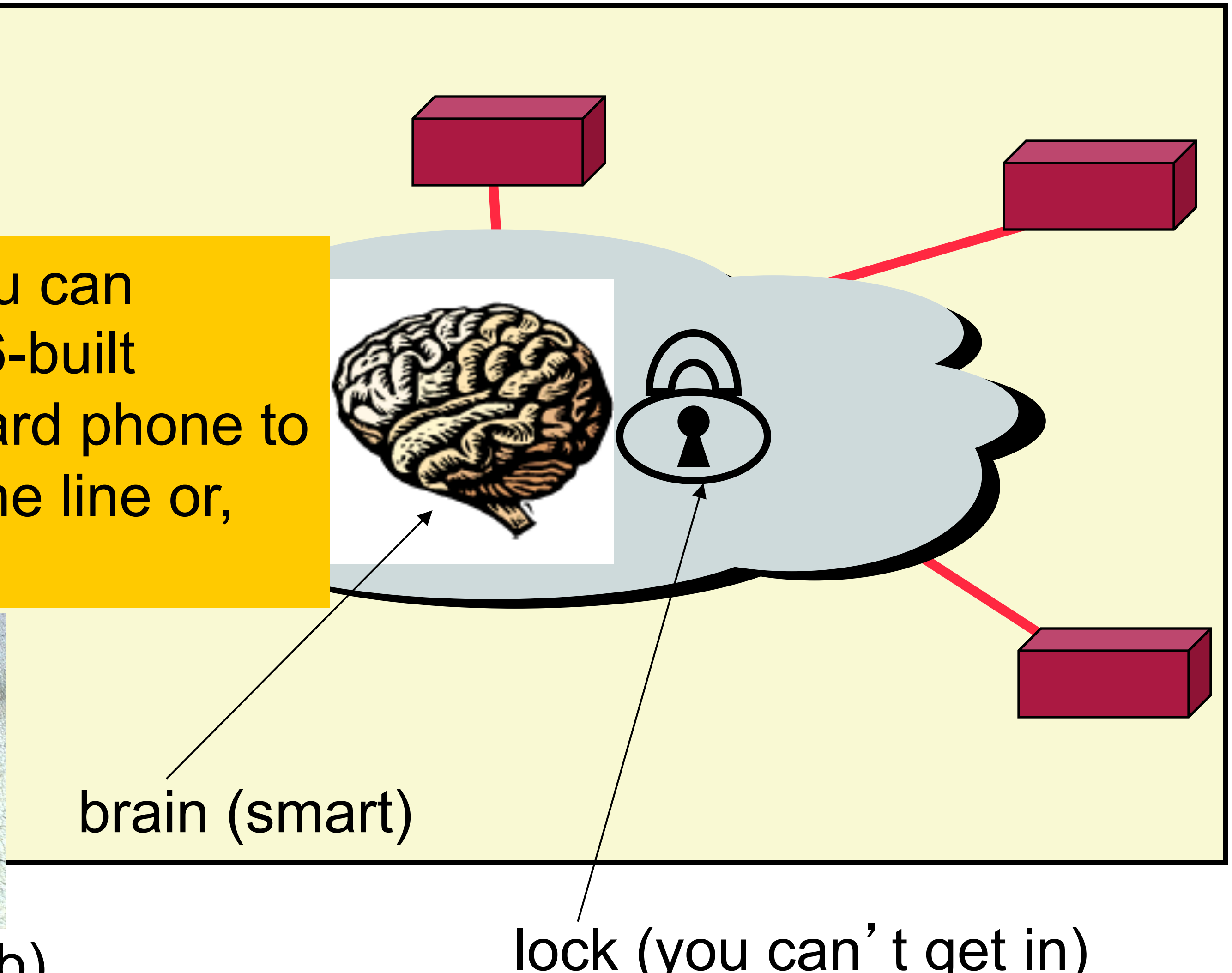
Chapter:
Internet Architecture

Structure

- Internet architecture
  - Requirements, assumptions
  - Design decisions
- Shortcomings and „Future Internet“ concepts
  - „Legacy Future Internet“: IPv6, SCTP, …
  - Security
  - QoS, multicast
  - Economic implications, „tussle space“
  - Mobility and Locator–ID split
  - In-network congestion control
  - Modules instead of layers
  - Delay-tolerant/disruption-tolerant networking
  - Content-based networking/Publish–subscribe architectures
  - Evolutionary vs. Revolutionary/Clean-slate

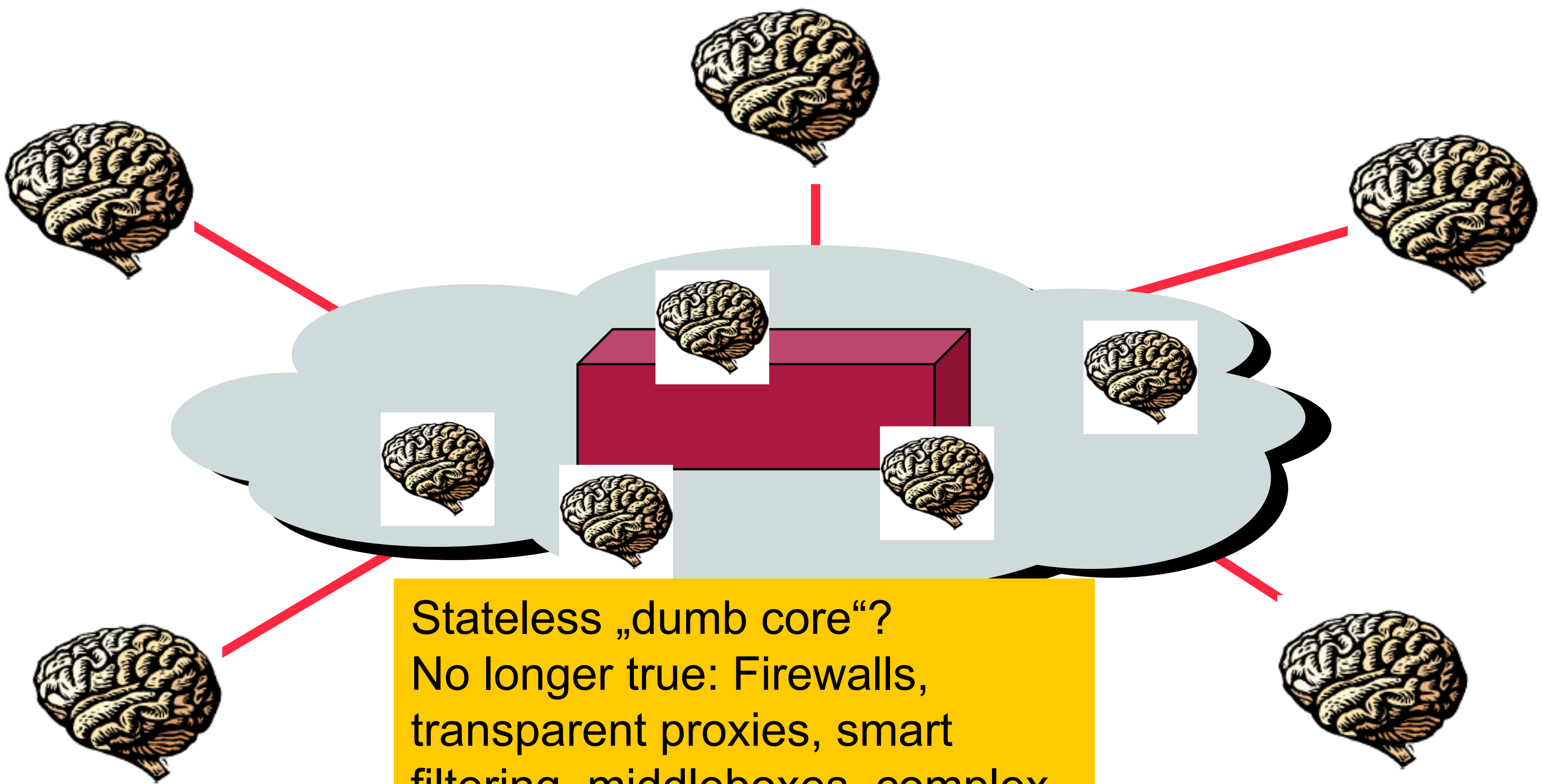
Common View of the Phone Network

Even today, you can connect a 1936-built German standard phone to an analog phone line or, e.g., a FritzBox



brick (dumb)

Common View of the IP Network



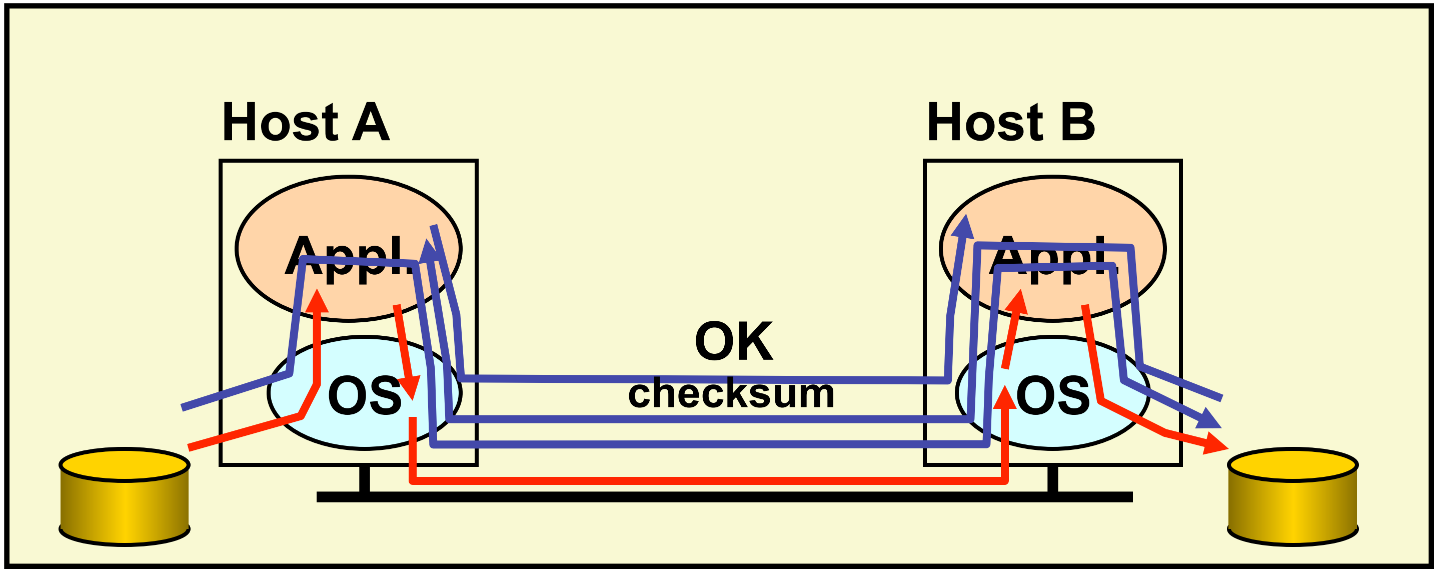
Stateless „dumb core“?
No longer true: Firewalls, transparent proxies, smart filtering, middleboxes, complex routing protocols like BGP, …

The Internet End-to-End principle

Internet End-to-End Principle

- "…functions placed at the lower levels may be *redundant* or of *little value* when compared to the cost of providing them at the higher level…"
- "…sometimes an *incomplete* version of the function provided by the communication system (lower levels) may be useful as a *performance enhancement*…"
- This leads to a philosophy diametrically opposite to the telephone world of dumb end-systems (the telephone) and intelligent networks.

Example: Reliable File Transfer



- Solution 1: make each step reliable, and then concatenate them
- Solution 2: each step unreliable – end-to-end check and retry

Discussion

- Is solution 1 good enough?
  - No – what happens if components fail or misbehave (bugs)?
- Is reliable communication sufficient?
  - No – what happens in case of, e.g., disk errors?
- so need application to make final correctness check anyway
- Thus, full functionality can be entirely implemented at application layer; *no* need for reliability at lower layers

Discussion

Q: Is there any reason to implement reliability at lower layers?

A: YES: "easier" (and more efficient) to check and recovery from errors at each intermediate hop

- e.g.: faster response to errors, localized retransmissions

Internet Design Philosophy (Clark' 88)

David Clark: The design philosophy of the DARPA internet protocols
ACM SIGCOMM '88 Symposium proceedings on Communications Architectures and Protocols, pp. 106-114

In order of importance (note: Different ordering of priorities would result in different architecture)

0 Connect existing networks
   initially: ARPANET, ARPA packet radio, packet satellite network
1. Survivability
   ensure communication service even with network and router failures
2. Support multiple types of services
3. Must accommodate a variety of networks
4. Allow distributed management
5. Allow host attachment with a low level of effort
6. Be cost effective
7. Allow resource accountability

1. Survivability

- Continue to operate even in the presence of network failures (e.g., link and router failures)
  - As long as network is not partitioned, two endpoints should be able to communicate
  - Any other failure (excepting network partition) should be transparent to endpoints
- Decision: maintain end-to-end transport state only at end-points
  - eliminate the problem of handling state inconsistency and performing state restoration when router fails
- Internet: stateless network-layer architecture
  - No notion of a session/call at network layer
- Remark: "Internet was built to survive global thermonuclear war" = urban legend; untrue

2. Types of Services

- Add UDP to TCP to better support other apps
  - e.g., “real-time” applications
- Arguably main reason for separating TCP, IP
- Datagram abstraction: lower common denominator on which other services can be built
  - Service differentiation was considered (ToS bits in IP header), but this has never happened on the large scale

3. Variety of Networks

- Very successful (why?)
  - Because of minimalism
  - Only requirement from underlying network: to deliver a packet with a "reasonable" probability of success
- …but does *not* require:
  - Reliability
  - In-order delivery
  - Bandwidth, delay, other QoS guarantees
- The mantra: IP over everything
  - Then: ARPANET, X.25, DARPA satellite network, phone lines, …
  - Today: Ethernet, DSL, 802.11, GSM/UMTS, …
  - Soon: LTE, WIMAX, …

Other Goals

- Allow distributed management
  - Administrative autonomy: IP interconnects networks
    - each network can be managed by a different organization
    - different organizations need to interact only at the boundaries
    - … but this model complicates routing
- Cost effective
  - sources of inefficiency
    - header overhead
    - retransmissions
    - routing
  - …but "optimal" performance never been top priority

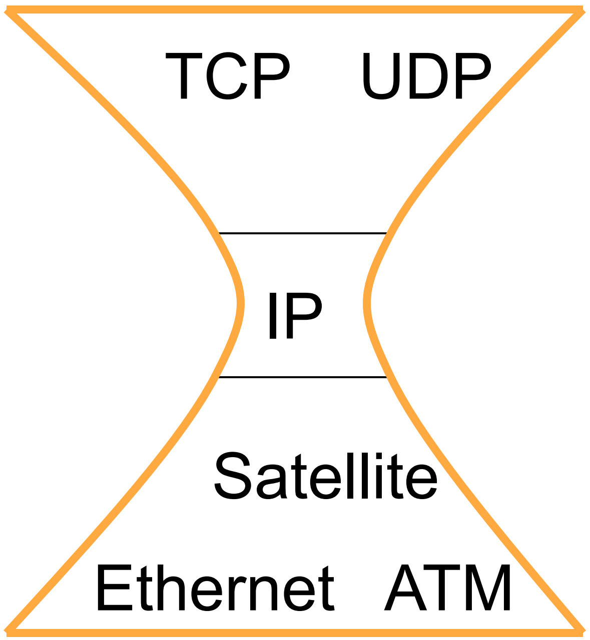
Other Goals (Cont)

- Low cost of attaching a new host
  - not a strong point → higher than other architecture because the intelligence is in hosts (e.g., telephone vs. computer)
  - bad implementations or malicious users can produce considerably harm (remember fate-sharing?)

- Accountability
  - Not a strong point: no financial interests (research network!)

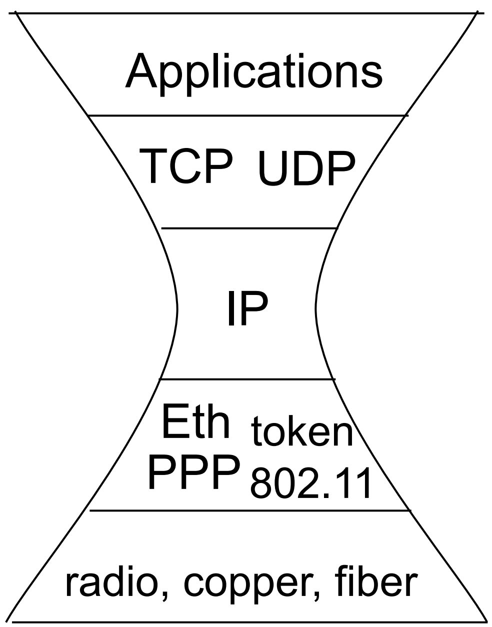
Summary: Internet Architecture

- Packet-switched datagram network
- IP is the glue (network layer overlay)
- IP hourglass architecture
  - All hosts and routers run IP
  - IP hides transport/application details from network
  - IP hides network details from transport/application
- Stateless architecture
  - No per-flow state inside network
  - Intelligence (i.e., state keeping) in end hosts, but not in core

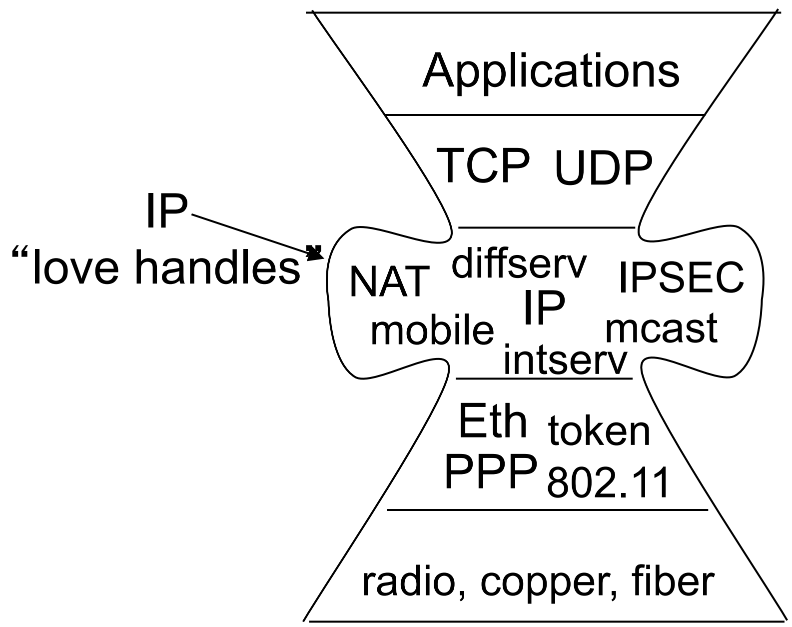


IP hourglass

Big picture: supporting new applications – losing the IP hour glass figure?



IP "hourglass"



Middle-age IP "hourglass"?